Attorney Docket No. 6770P001

APPLICATION FOR UNITED STATES LETTERS PATENT

For

# ADAPTIVE CLOCK RECOVERY

Inventors:

Martin Raymond Scott

Nicholas Faithorn

Timothy Michael Edmund Frost

Prepared by:

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, CA 90025
(408) 720-8300

"Express Mail" mailing label number: EV33658125US
Date of Deposit: August 22, 2003
I hereby state that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-14550

Christopher P. Marshall
(Typed or printed name of person mailing paper or fee)
C.P. Marshall
(Signature of person mailing paper or fee)
8/22/2003
(Date signed)

## Adaptive Clock Recovery

The invention relates to the recovery of clock signals for a TDM output from packets of TDM data which have been transmitted over a packet network.

TDM links are synchronous circuits, with a constant bit rate governed by the service clock $f_{service}$. With a packet network the connection between the ingress and egress frequency is broken, since packets are discontinuous in time. From Figure 1, the TDM service frequency $f_{service}$ at the customer premises must be exactly reproduced at the egress of the packet network ($f_{regen}$). The consequence of a long-term mismatch in frequency is that the queue at the egress of the packet network will either fill up or empty, depending on whether the regenerated clock is slower or faster than the original. This will cause loss of data and degradation of the service.

The relevant standards on circuit emulation services over ATM, ITU standard I.363.1 and ATM Forum standard af-vtoa-0078 refer to the concept of adaptive clock recovery in general terms.

According to the invention there is provided a method of recovering a clock signal, and a clock recovery system, as set out in the accompanying claims.

Embodiments of the invention will now be more particularly described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a schematic diagram showing a leased line TDM service being carried across a packet network; and

Figure 2 is a schematic diagram showing a clock recovery method based on Transit Time in accordance with an embodiment of the invention.

In Figure 1, the rate of transmission of packets from the source device is isochronous and determined by $f_{service}$. However, the rate of packet arrival at the destination device is perturbed by the intervening packet network. Packets will typically arrive in bursts separated by varying amounts of delay. The delay between successive packets and bursts will vary depending on the amount of traffic in the network. The characteristics of the network are non-deterministic, but over the long term the rate of arrival at the destination will equal the rate of departure at the source (assuming no lost or duplicate packets).

The TDM output at the destination is isochronous and determined by $f_{regen.}$ This is provided by the Digitally Controlled Oscillator (DCO) (22) in Figure 2. The output is supplied from a Packet Delay Variation (PDV) Buffer (12). If the buffer has zero packets in it when the TDM output requires to transmit then an underrun will occur, which is undesirable. In order to minimise underrun events it is necessary to build up the PDV buffer (12) so that it contains sufficient packets to supply the TDM output for the majority of inter packet delays. However, the PDV buffer (12) cannot be made arbitrarily large because this directly increases the end to end latency which, in general, is required to be as low as possible, the maximum tolerable latency being dependent on the application. For example, voice requires lower latency than data.

Thus the optimal PDV Buffer depth depends upon network conditions and application. The clock recovery method described here allows the buffer depth to be varied independently of the clock recovery mechanism. This allows the clock recovery to stabilise prior to setting up the PDV Buffer, and allows the buffer to be changed during operation to match any underlying shift in network characteristics.

When a packet arrives at the Packet Input (10) it is placed into a PDV Buffer (12) in a Queue (14). It also has its timestamp extracted and passed to a Differencer (16). The Remote timestamp is determined at the source device when the packet is created, by counting periods of the source TDM clock $f_{service}$ shown in Figure 1. The Local Timestamp is determined, when the packet is received, by counting periods of the local TDM clock $f_{regen}$ also shown in Figure 1.

The Differencer (16) subtracts the Remote Timestamp from the Local Timestamp to obtain the Transit Time.

Transit Time (n) = Local Timestamp(n) – Remote Timestamp(n)
*Equation (1)*

It should be noted that because the Local and Remote clock frequencies and initial counts are not initially synchronised with respect to each other, the quantity "Transit Time" in Equation 1 does not represent the actual time that the packet has taken to travel between the source and destination but is instead a measure of the difference between counts of the bit clock cycles of the two clocks.

Hence, given an ideal fixed delay packet network, the Transit Time value will decrease if $f_{service}$ exceeds $f_{regen}$ , will increase if $f_{regen}$ exceeds $f_{service}$ , and will remain constant if the frequencies are identical. In one embodiment of the invention, the counter 30 used to determine the local timestamp is initialised to the first received value of the remote timestamp, setting the initial value of transit time to zero.

For each received packet (assuming no lost packets), the Remote Timestamp will increase by a fixed amount which represents the number of bit clock cycles at the source that have elapsed since the previous packet i.e. the number of bits contained in the packet payload.

This system is robust in the presence of lost packets because the remote and local timestamps of the next packet received following the lost packet(s) are unaffected by the loss. The lost packets merely represent a short term loss of resolution in the measurement. In a typical system there will be thousands of packets per second so that a packet loss rate which is at the maximum likely to occur (i.e. a few percent) will have a negligible effect on the algorithm.

The Timestamps are counts (rather than actual times) that can be thought of as the phase of the clock.

So Equation (1) above can be represented as:

$$\Phi(n) = \theta_2(n) - \theta_1(n)$$

Where:

$\Phi(n)$ is the Transit Time

$\theta_2$ is the phase of the Local Clock in bit periods

$\theta_1$ is the phase of the Remote Clock in bit periods

An indication of the frequency difference, can be obtained:

$$\Delta\Phi(n) = \Phi(n) - \Phi(n-1)$$
$$= [\theta_2(n) - \theta_1(n)] - [\theta_2(n-1) - \theta_1(n-1)]$$
$$= [\theta_2(n) - \theta_2(n-1)] - [\theta_1(n) - \theta_1(n-1)]$$
$$= [f_2(n) - f_1(n)].\ \Delta t$$
$$= \Delta f.\ \Delta t$$

Where

$\Delta t$ is the interval between packet arrivals.

f1 is the Remote clock frequency

f2 is the Local clock frequency

$\Delta f$ is the difference between the local and remote clock frequencies

$\Delta f$ may be used by a Clock Control Algorithm (20) to adjust the Local Clock frequency.

With a real network the Transit Time, $\Phi(n)$, fluctuates due to the burst nature of the incoming packet stream. This causes fluctuations in the recovered clock. Therefore a filter function (24) is provided, which provides the following benefits:

- reduces the workload of the Clock Control Algorithm (20) (which may be implemented by an external CPU) in terms of numerical processing
- reduces the workload of the Clock Control Algorithm (20) by allowing the Clock Control Interval to be increased

- reduces fluctuations in the recovered clock

For example, the filter (24) may be a first order low pass filter with the following difference equation that is simple to implement in hardware without requiring any dividers or multipliers:

$$\Phi_{av}(n) = \Phi_{av}(n-1) + (\Phi(n) - \Phi_{av}(n-1))/2^{P} \qquad \textit{Equation (2)}$$

Where:

$\Phi_{av}$ is the Filter Output

$\Phi$ is the Difference value

P is a programmable parameter that determines the time constant of the filter

n is the sample number that increments each time that a packet is received at the Packet Input

The Clock Control Algorithm (20) can read $\Phi_{av}$ at a fixed interval, to obtain

$$\Delta\Phi_{av}(m) = \Phi_{av}(m) - \Phi_{av}(m-1)$$

It can be shown that

$$\Phi_{av}(m) = \theta_{2\,av}(m) - \theta_{1\,av}(m)$$

Where $\theta_{2\,av}(m)$ and $\theta_{1\,av}(m)$ are also obtained by the filtering as described by Equation (2)

Therefore

$$\Delta\Phi_{av}(m) = [\theta_{2\,av}(m) - \theta_{2\,av}(m-1)] - [\theta_{1\,av}(m) - \theta_{1\,av}(m-1)]$$

$$\Delta\Phi_{av}(m) = [f_2(m) - f_1(m-1)].\ \Delta t$$

$$= \Delta f.\ \Delta t$$

Where

$\Delta t$ is the interval between samples (m) and (m-1).

f1 is the Remote clock frequency

f2 is the Local clock frequency

$\Delta f$ is the difference between the local and remote clock frequencies

The Clock Control Algorithm (20) will read $\Phi_{av}$ and determine the correction required to converge the Local clock to the Remote clock, and write the required Frequency to a DCO (22).

A simple first order Clock Control Algorithm is given by the following difference equation:

$$F(m) = F(m-1) + \beta\, (\Delta\Phi_{av}(m) / \Delta t)$$

Where:

$F(m)$ is the Frequency to be written to the DCO;

$F(m-1)$ is the Current DCO Frequency;

$\beta$ is a constant that determines a time constant;

$\Delta\Phi_{av}(m)$ is the change in the average transit time;

m is the sample number that increments each time the Clock Control Algorithm reads the value of $\Delta\Phi_{av}$; and

$\Delta t$ is the time interval between reads of values by the Clock Control Algorithm.

The effect of this Clock Control Algorithm is to limit the frequency difference between the remote and local clocks, by correcting the DCO frequency in response a change in the filtered value of transit time, $\Delta\Phi_{av}(m)$. The constant term $\beta$ determines the time constant of this correction. It is selected to track long term drift in the remote clock frequency, $f_{service}$, but reject short term variation due to packet delay variations.

An enhanced Clock Control Algorithm is given by the following equation:

$$F(m) = F(m-1) + \beta\, (\Delta\Phi_{av}(m) / \Delta t) + \gamma\, ((\Phi_{av}(m) - K) / \Delta t)$$

Where:

$F(m)$ is the Frequency to be written to the DCO;

$F(m-1)$ is the Current DCO Frequency;

$\Delta\Phi_{av}(m)$ is the change in the average transit time;

m is the sample number that increments each time the Clock Control Algorithm reads the value of $\Delta\Phi_{av}$;

$\beta$ and $\gamma$ are constants that determine time constants of the algorithm;

K is a constant that provides a "centre value" for the filtered transit time, $\Phi_{av}(m)$; and

$\Delta t$ is the time interval between reads of values by the Clock Control Algorithm.

The difference between this equation and the simple Clock Control Algorithm is the addition of the term $\gamma\ ((\Phi_{av}(m) - K) / \Delta t)$. This term tends to keep the filtered value of the transit time $\Phi_{av}(m)$ to a centre value K, by correcting the DCO frequency when $\Phi_{av}(m)$ drifts away from the centre value. This has the effect of controlling phase shift between the remote and local clocks. Since the initial value of transit time $\Phi(1)$ is normally set to zero, the value of K used is also normally set to zero.

The constant term $\gamma$ determines the time constant of this correction. As with the $\beta$ term, this is selected to track long term drift in the remote clock frequency but reject short term variation due to packet delay variations

A PDV Depth Control Algorithm (26) makes relatively infrequent adjustments to the PDV Buffer (12) (by adding or removing packets to or from the buffer) which may be based on :

- Filtered Depth reading of queue depth provided by filter (28), which may be of the type described by Equation (2)
- Underrun events (indicating the Queue (14) is too small)
- Maximum and Minimum Depth readings
- Network Delay Measurements (For example obtained by network "ping" utility)

The Minimum & Maximum depth values are reset to the current Queue Depth each time they are read by the PDV Buffer Depth Control Algorithm (26), and are then adjusted whenever the Packet Queue Depth is altered.

Alternative Filter algorithms may be used.

Alternative Clock Control Algorithms may be used e.g. 2nd and higher order, fuzzy logic, neural networks, and self-tuning algorithms that vary parameters such as the time constant or Clock Control Interval over time.

An internal or external CPU may be used for the Clock Control & Depth Control Algorithms

Fractions of a bit, byte, frame or packet counts may be used as the Timestamp units instead of bits.

The method as described makes use of all the data packets. It is also possible to use a subset of the packets, or to use special timing packets.

It would be possible to implement the method without remote timestamps where packets are of a consistent payload size, provided a sequence number is available in the packets transmitted from the source to the destination. The remote timestamp can be reconstructed at the destination by multiplying the sequence number by the size of the TDM payload. As mentioned above, the size may be expressed in fractions of a bit, bits, bytes, frames or packets. The use of a sequence number applied by the source device ensures that the timestamp calculation is not corrupted by lost or out-of-sequence packets.

The method has application in timing recovery over packet based systems or other asynchronous systems. A typical application of the method described above is in emulation of TDM (time division multiplexed) circuits across a packet network, such as Ethernet, ATM or IP. Circuit emulation may be used to support the provision of leased line services to customers using legacy TDM equipment. For example, Figure 1 shows a leased line TDM service being carried across a packet network. The advantages are that a carrier can upgrade to a packet switched network, whilst still maintaining their existing TDM business.

The clock recovery method described above provides the following advantages:

1. The method is able to make use of all of the incoming data packets at the destination device to converge average $f_{regen}$ (i.e. local clock) to average $f_{service}$ (i.e. remote clock).
2. No expensive Clock Generation Circuits are required (such as oven controlled crystal oscillators).
3. Timestamps contained in the incoming packets are compared to a Local Timestamp value to obtain a Transit Time value.
4. The sequence of Transit Time values that is obtained are filtered.
5. The filtered Transit Time value is used by a Clock Control Algorithm to adjust the Local TDM Clock of the device.
6. The separation of the filter from the Clock Recovery Algorithm allows the Clock Control Algorithm to operate at a much slower rate than the filter. So that, for example, a high speed filter could be implemented in Hardware and a low speed Clock Control Algorithm with an external CPU. This confers significant benefits, such as flexibility, reduction of development risk, ease of optimising the solution for a specific environment etc.
7. A method is provided to enable the Local Timestamp to be initialised to the value of the first received Remote Timestamp to minimise wraparound problems and prevent a start up error with the recovered clock.
8. A method is provided to allow packets to be deleted from the PDV Buffer and dummy packets to be inserted into the PDV Buffer in order to adjust the device latency. This does not affect the Local Timestamp value mentioned above.
9. The PDV Buffer Depth is filtered at an appropriate interval.
10. Minimum & Maximum PDV Buffer Depth values are maintained
11. The filtered PDV Buffer Depth, and Minimum & Maximum PDV Buffer Depth values may be used by a Buffer Depth Control Algorithm which may run at a much slower rate than the rate at which the filter is updating.
12. The clock recovery method described here allows the PDV Buffer depth to be varied independently of the clock recovery mechanism. This allows the clock recovery to

stabilise prior to setting up the PDV Buffer, and allows the buffer to be changed during operation to match any underlying shift in network characteristics.

It is also possible to exclude late packets from the Transit Time calculations, which may improve performance. Such packets may artificially increase the Transit Time, causing the recovered clock to appear as though it is running too fast.